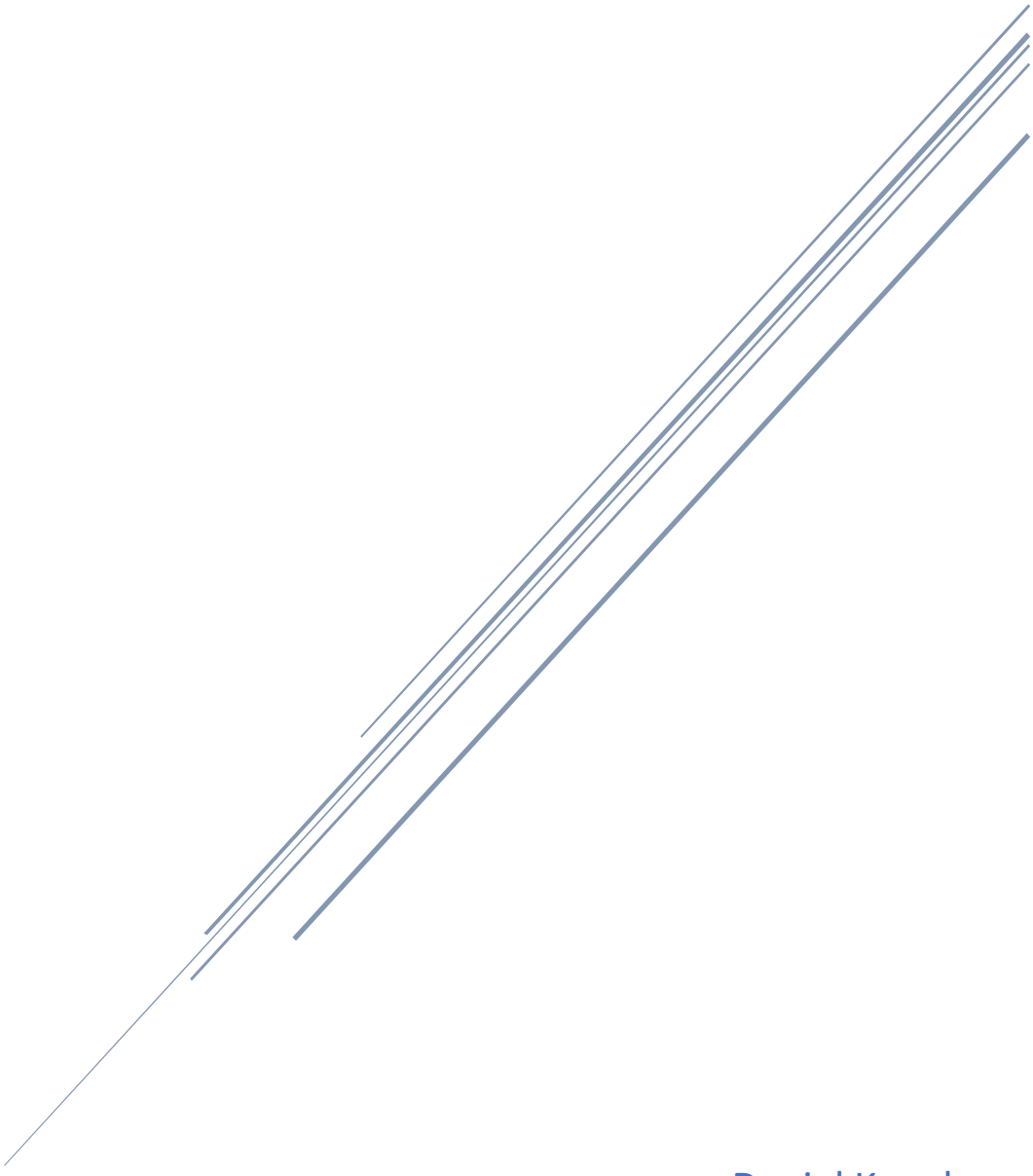
MOBILE APPLICATIONS DEVELOPMENT 3

Design & Storyboard for a 3D Rail Shooter Game



Daniel Keogh
G00335478

CONTENTS

1 INTRODUCTION

The following document specifies the design for a 3D Rail Shooter game developed with the Unity Game Engine. The game described in this document is a flying rail shooter loosely based on the game *After Burner,* except it will be set in a space environment.

1.1 GAME SUMMARY

In this game the player controls a spacecraft with their objective being to simply shoot down enemy ships while simultaneously evading their return fire.

2 RESEARCH

In this section I'll provide a summary of the research I conducted into two of the game categories permitted for this project.

2.1 3D RAIL SHOOTERS

A "Rail Shooter" is a type of shooter game characterised by the majority of the players movements being "on-rails", essentially meaning the path the player moves is predetermined [1]. The player may be able to move from side to side in order to dodge enemies or obstacles, but they are unable to control their overall destination. If there is any player choice at all as to where to go, it is typically in the form of branching paths initiated by specific "fork in the road" situations [1]. Rail shooters may be playable in either a first or a third person perspective, and the game's world needs to exist in three dimensions (or at least simulate the appearance of being in three dimensions [2]), otherwise the game would be better categorised as a vertical or horizontal scroller [3]. Examples of third-person rail shooter games include the *After Burner* series, and *Star Fox*.

Figure 1: After Burner Climax

2.2 3D Endless Runners

The "Endless Runner" is a genre in which players must dodge obstacles as they automatically and continuously scroll onto the screen, with the goal of not crashing into anything for as long as possible [4]. These games may be presented in 2D, like *Jetpack Joyride,* or in 3D like *Crash Bandicoot* and *Temple Run*. Difficulty is usually denoted by an increase in the speed of gameplay as the player progresses. As the game speeds up, the player's character appears to run faster and faster as if on a giant treadmill. Obstacles approach faster and faster, requiring faster reactions from the player [5].

By design these games are *endless* and consist of a single, continuous level with the long-term goal being to achieve a better high-score on the next playthrough [4]. However, a game can still fall under the category of "Endless Runner" without being strictly endless. For instance, *Rayman Jungle Run* is an example of a level-based, endless runner with a clear end state [6].

Infinite running games have found particular success on mobile platforms as such devices are well-suited to the small set of controls these games require [7]. Game controls typically consist of swiping left/right for horizontal movement, swiping up to jump and swiping down to slide under obstacles. Some incorporate "tilt-control" to move the game character from left to right [5].

Endless runner games often make use of *Procedural Generation* instead of using pre-built levels in order to add variety and create levels that are unique on each playthrough [8].

Figure 2: Temple Run

2.3 THIS GAME

As mentioned above, this game is a third person, 3D rail shooter. The direction the player moves will be predetermined, but they will have the ability to tilt their ship up, down, left or right in an effort to evade enemy projectiles. Although rail shooters are arguably less well suited to mobile devices than an endless runner, this game will attempt to rectify this issue by using the device's accelerometer for movement when it is run on a mobile device.

3 GAMEPLAY & STORYBOARD

3.1 FRONTEND

3.1.1 Splash Screen

When the game launches, the user will be first greeted with the default "Made with Unity" splash screen. They will then automatically transition to the game's main menu screen.

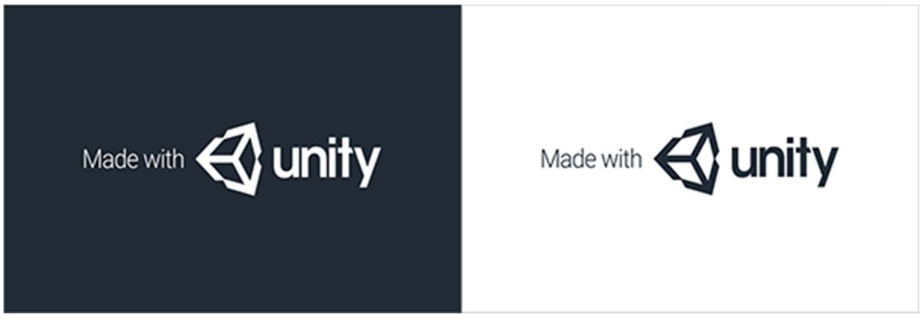


Figure 3: Default Unity Splash Screen in dark & light format.

3.1.2 Start Menu

The game will use a simple menu system for selecting options. The main/start menu will contain four options:

- *Play* - Launches the game in single-player.
- *Multiplayer* - Launches the game in multi-player.
- *Settings* - Displays the game's options menu, where the user can alter settings like the volume of the game's music, enable/disable sound effects and any other relevant settings.
- *Quit* - Exits the game.

Figure 4: Main Menu

3.1.3 Game Over Menu

The game over menu is displayed whenever the player dies or completes all levels in the game. It should display the player's final score for this playthrough and show how that score compares to their overall high-score. This menu should also provide a way for the player to either play the game again, or be directed to the start menu described above.

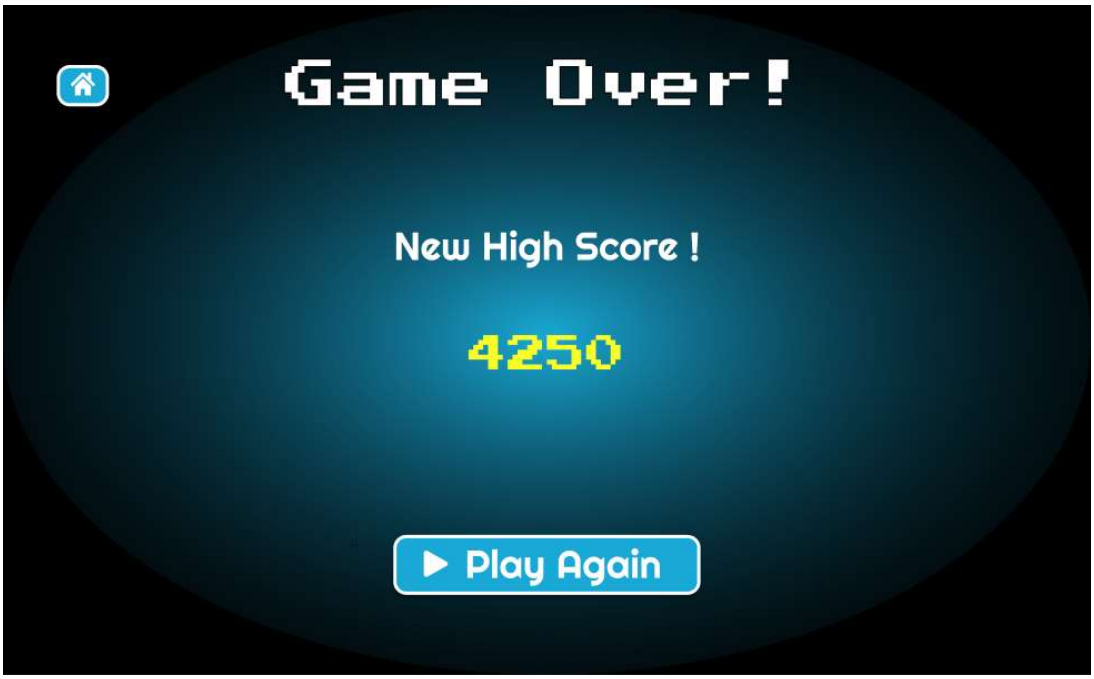


Figure 5: Game Over menu

3.2 In-Game Menus

3.2.1 Pause Menu

The pause menu will consist of a semi-transparent overlay on top of the game screen containing the below options:

- *Resume* - Resuming the game.
- *Quit* - Exit back to the main menu.

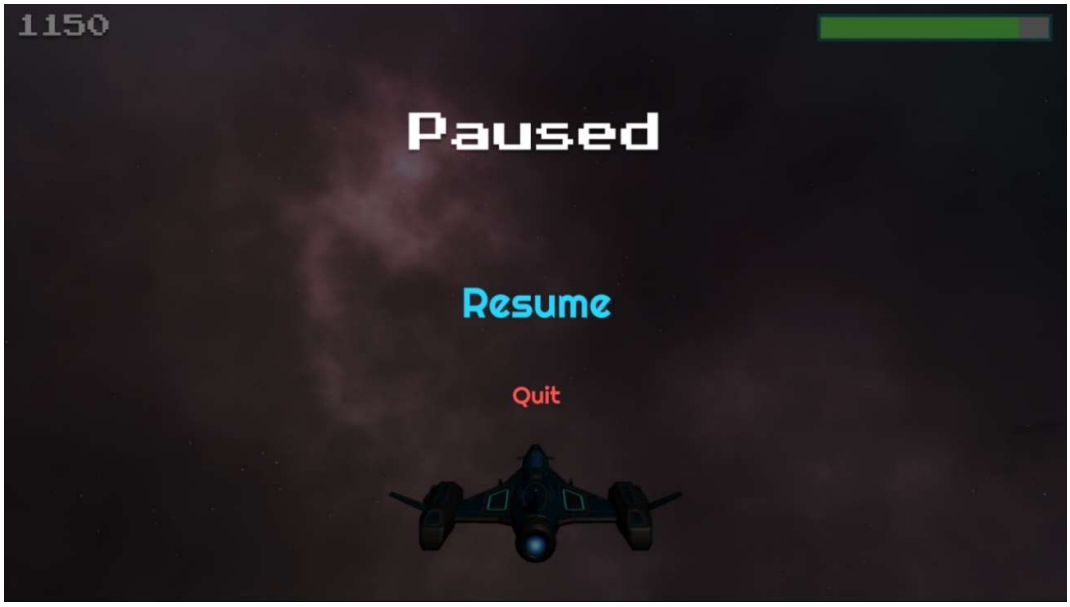


Figure 6: Pause menu

3.2.2 Heads-Up Display (HUD)

The HUD should contain the following pieces of information:

- A number indicating the player's current score.
- A health bar indicating how much health the player has remaining. This should change in appearance in order to indicate if the player's health is low.
- A crosshair/reticle indicating what the player is aiming towards.
- A button for shooting at enemies. This need only be visible when the game is run on a mobile device, as on desktop devices the spacebar is available for use instead.

Figure 7: HUD [9]

3.3 Control Mechanisms

3.3.1 PC, Mac, Linux Standalone

The controls for PC are pretty much what ought to be familiar to most players, with the arrow keys and WASD keys controlling movement. The controls are illustrated in the below table.

Key(s)	Function
W ↑	Tilt the ship downwards.
A ←	Tilt the ship to the left.
S ↓	Tilt the ship upwards.
D →	Tilt the ship to the right.
Space	Shoot a single time. Press and hold to continuously shoot.
esc	Pause or resume the game.

3.3.2 Mobile

For mobile it might be best to force the game to be displayed in a landscape (16:9) orientation and use the device's accelerometer to control the player's movement. Shooting can be controlled by detecting taps on a button located on the user's screen (*see Figure 7*).

3.4 The Game

The game begins with the player flying alone in the centre of the screen. Waves of enemies will then begin to slowly spawn into the scene and shoot lasers directed at the player. Progression to the next level of play is achieved by surviving each wave of enemies that are spawned over the course of the level. When a player has successfully progressed to the next level, this is indicated to them via a brief popup over the scene (*Figure 9)*.

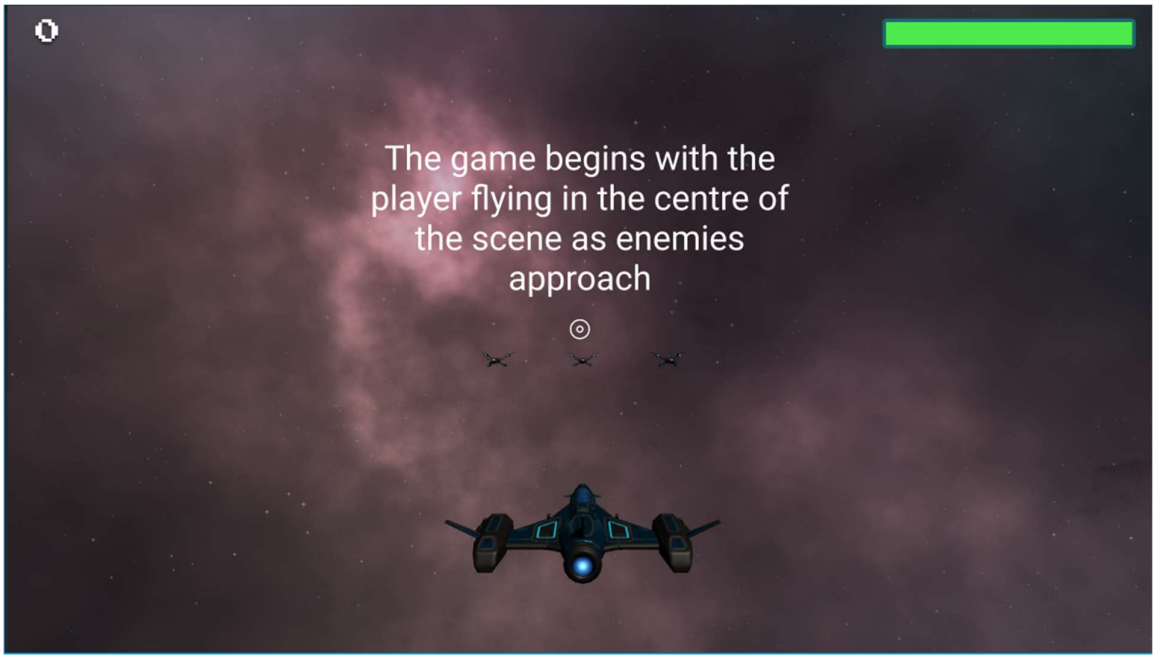


Figure 8: Initial setup

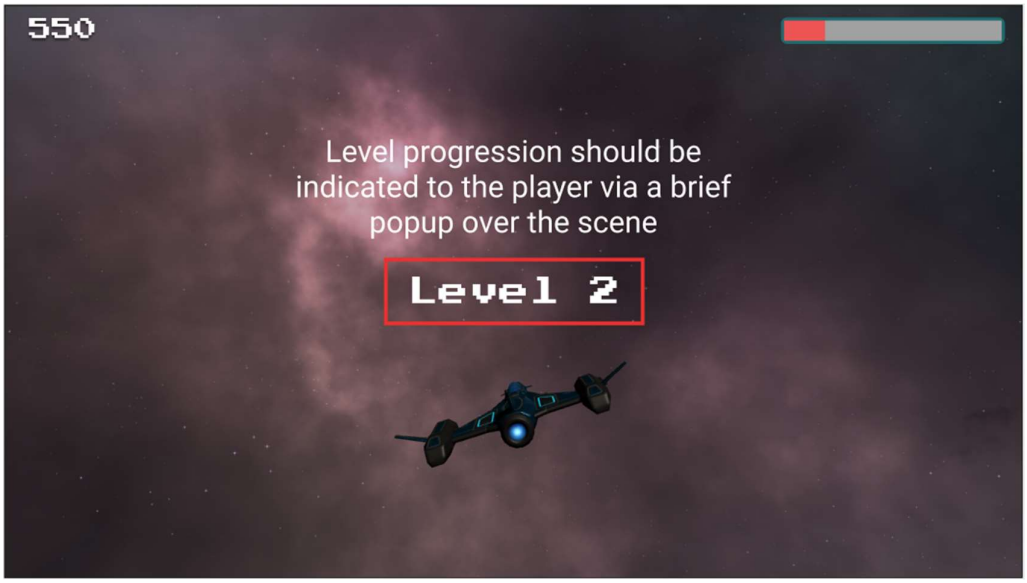


Figure 9: Level progression

3.5 Level Design

The game will be composed of three levels. In each level, waves of enemy ships will approach the player's position and attempt to shoot them down. The increase in difficulty from one level to the next is mainly denoted by an increase in the number of waves per level, as well as an increase in the quantity of enemies in each wave. However, the exact number of enemies/waves in each level might best be resolved during playtesting so as to better determine the appropriate level of difficulty for the player. Likewise, the number of hits the enemy & player ships can take before being destroyed is left to the discretion of the developer.

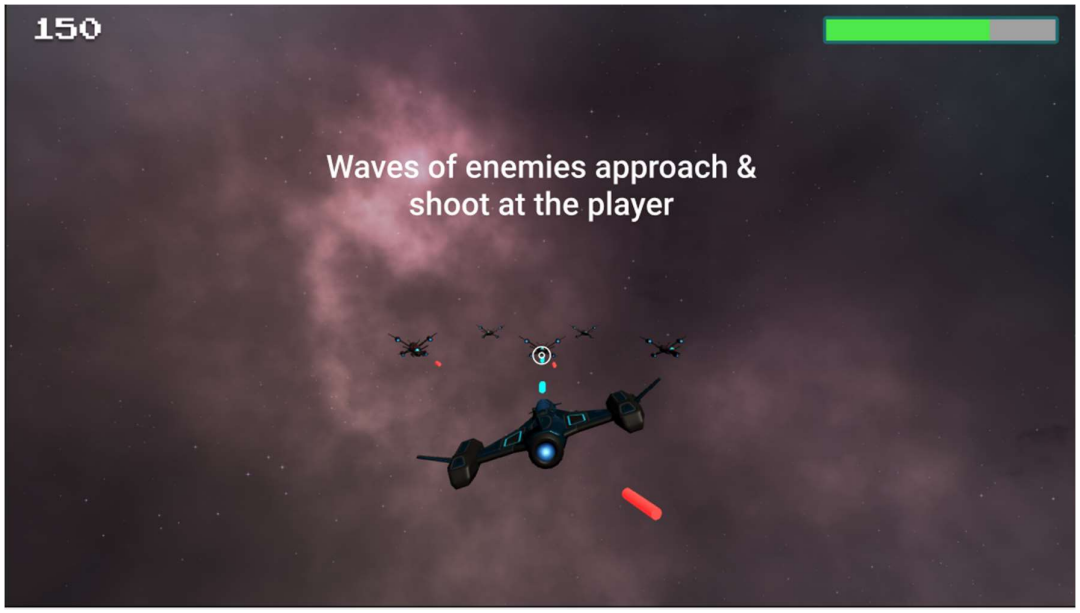


Figure 10: Enemy wave

Although all the sketches included in this document show an empty space scene, the game would ideally use Unity's terrain system in order to provide a more interesting environment for the player, similar to *Figure 1*. For instance, the scene below depicts the player flying over a barren moonscape.

Figure 11: Level with terrain

3.6 Enemies/Targets

There will be two types of enemies in the game. The main difference between them will be their size and speed of movement. The smaller enemy ships will be harder to shoot, and the player will therefore be awarded more points for destroying them.

Object	Points Awarded
Normal enemy ship	50
Smaller enemy ship	100

3.7 Collectibles/Power ups

There will be two types of pickups in the game. Pickups appear sporadically and the exact rate at which they are available is at the discretion of the developer. Pickups can be acquired by shooting the pickup's sprite.

- *Health pickup* - Replenishes the player's health.

- *Shield pickup* - Provides the player with some temporary immunity from damage. Player immunity should either expire after a certain amount of shots have been blocked by the shield, or after a given duration (whichever comes first).

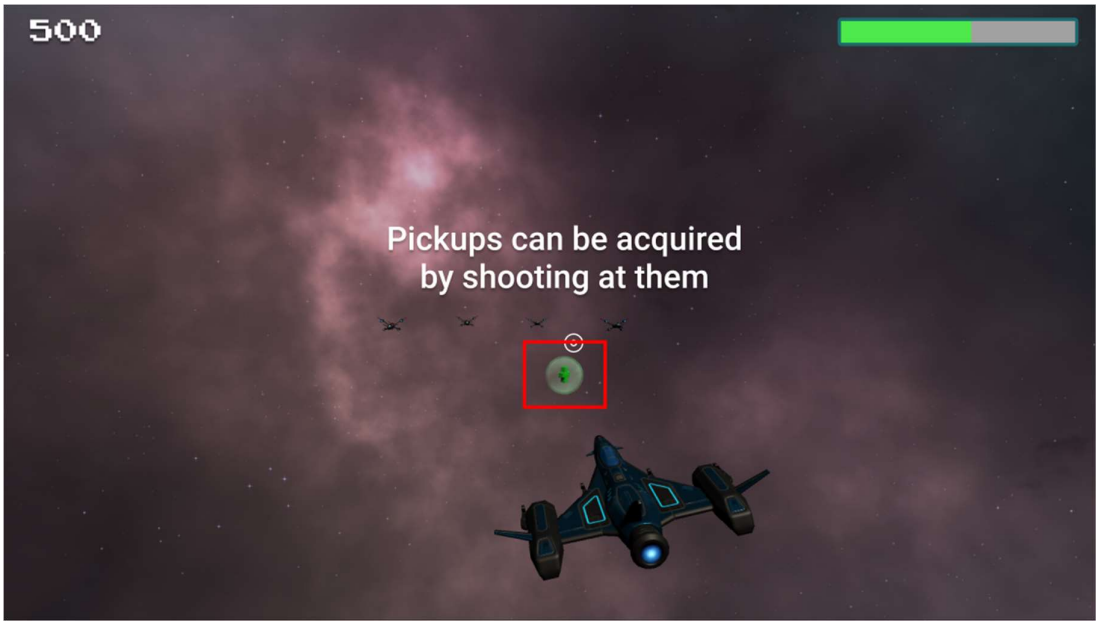


Figure 12: Power ups

If the user acquires a power-up there should be some sort of auditory cue indicating so. In the case of the shield pickup, it should also be indicated by altering the health bar in some way (e.g. changing its colour to blue). The health bar can be gradually changed back to its original colour as the shield wears off.

3.8 Multiplayer Functionality

When playing in multiplayer, players interact by cooperating alongside each other. If one of the players die, they will not be respawned until the beginning of the next level. If both players die in the same level, the game over screen should be displayed.

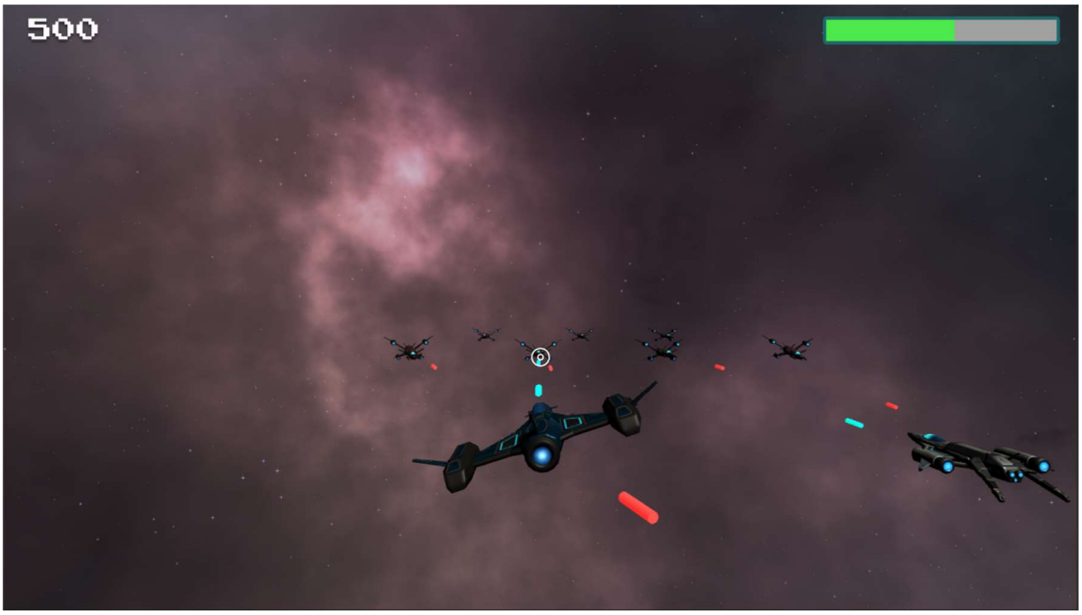


Figure 13: Multiplayer

4 Suggested Assets

In this section I'll list some assets from the Unity Asset Store that could be suitable for use in the game. All the game images included in this document use sprites contained in the below asset packs.

- "Star Sparrow Modular Spaceship" pack by *Ebal Studios* - Contains sprites that could be used for the player & enemy ships.
  https://assetstore.unity.com/packages/3d/vehicles/space/star-sparrow-modular-spaceship-73167

- "PowerUp particles" pack by *MHLab* - Contains sprites that could be used for the player health & shield power-ups.
  https://assetstore.unity.com/packages/vfx/particles/powerup-particles-16458

4.1 Skybox

- Something other than the default Unity skybox should be used in order to provide a space-themed backdrop for the game. E.g. The "Starfield Skybox".
  https://assetstore.unity.com/packages/2d/textures-materials/sky/starfield-skybox-92717

4.2 Audio

- UI sound effects could be achieved using the "UI Sfx" pack.
  https://assetstore.unity.com/packages/audio/sound-fx/ui-sfx-36989

4.3 Fonts

The fonts used in the above game sketches are "Righteous" & "Press Start 2P". The icons are from "Font Awesome". These fonts don't necessarily have to be used, but they're included here for the sake of completeness.

4.4 Other

- UI components like the health bar and menus can be done manually using the Unity UI system.

- Explosions and other effects (thrusters on the player & enemy ships etc.) could also be done manually using *ParticleSystem* components.

- Player & enemy lasers are relatively easy to create using the built-in 3D Objects.

5 REFERENCES

1. “Rail Shooter”, TV Tropes, Accessed 10/10/2020. https://tvtropes.org/pmwiki/pmwiki.php/Main/RailShooter

2. 2.5D, Wikipedia, Accessed 10/10/2020. https://en.wikipedia.org/wiki/2.5D

3. “Top 10 Rail Shooters of All Time”, Modern Classic, 23/08/2018. https://youtu.be/bIo2X7t0zA4?t=51

4. “Endless Running Game”, TV Tropes, Accessed 10/10/2020. https://tvtropes.org/pmwiki/pmwiki.php/Main/EndlessRunningGame

5. “Endless Runner Games: Evolution and Future”, Jerry Momoda. http://jerrymomoda.com/analysis-endless-runners/

6. “Endless Runner”, Giant Bomb, Accessed 10/10/2020. https://www.giantbomb.com/endless-runner/3015-7179/

7. “Platform Game – Endless Running Game”, Wikipedia, Accessed 10/10/2020. https://en.wikipedia.org/wiki/Platform_game#Endless_running_game

8. “Procedural Generation”, Wikipedia, Accessed 10/10/2020. https://en.wikipedia.org/wiki/Procedural_generation#Video_games

9. The icon shown in *Figure 7* was taken from this URL: https://www.flaticon.com/free-icon/missile_2590496